Team Vivek Dhayaal at SemEval-2026 Task 13 Subtask B: Multi-Class Authorship Detection

Vivek Dhayal
vivekdhayaal@gmail.com

Abstract

The widespread adoption of Large Language Models (LLMs) has accelerated significant progress in code generation while simultaneously giving rise to concerns related to academic honesty, software security, and the attribution of authorship. Traditional detection tools often fail when faced with diverse programming languages and modern LLMs.

We introduce GPAD (Generative-Process-Aware Detection), an extension of CodeBERT that augments the attention mechanism with token-level entropy signals and stylometric features, improving robustness across generators and domains.

Our Solution is designed for the SemEval-2026 Task 13 on detecting machine-generated code under diverse conditions by evaluating generalization to unseen languages, generator families, and code application scenarios.

1 Introduction

The key challenge is distinguishing human-written from AI-generated code, as modern LLMs produce realistic code. Early models like CodeBERT built strong code–language understanding, while newer systems such as Codet-m4 (2025) enhanced robustness across languages, generators, and domains. The main ongoing difficulties are zero-shot detection—identifying code from unseen AIs—and defining "human-ness" in code. While approaches like Approximated Task Conditioning (ATC) attempt to tackle this by analyzing whether a code's structure and logic align with its intended task, these methods remain imperfect and the problem largely unsolved.

2 Related Work

As LLM-generated code became more ubiquitous and sophisticated, the research focus quickly shifted toward building robust, generalized detection systems capable of handling real-world diversity. The Codet-m4 framework (Orel and Others, 2025) emerged as a comprehensive approach designed to perform reliably across multilingual, multi-generator, and multi-domain settings. This work demonstrated that effective detection must incorporate feature engineering—such as abstract syntax tree (AST) depth and complexity metrics—alongside PLMs to prevent models from being fooled by variations in programming language or application scenario. This need for comprehensive evaluation was also formally recognized by community efforts like SemEval-2026 Task 13 (Organizers, 2026), which set new benchmarks for testing detector generalization.

The initial wave of research focused on establishing a strong computational understanding of source code, which is fundamentally different from natural language text. Foundational work by Feng et al. (Feng et al., 2020) introduced CodeBERT, a pre-trained model trained on a massive dataset of code-text pairs. This bimodal training allowed the model to grasp the semantic and structural relationship between programming logic and human intent, providing the core representation needed for subsequent code analysis tasks. Early detection methods often leveraged these pre-trained language models (PLMs) by fine-tuning them on simple binary classification tasks, attempting to distinguish code based on basic statistical artifacts or low-entropy tokens inherent to AI generation.

3 Architecture Diagram

The model integrates entropy-weighted attention inside CodeBERT to sharpen token focus, while a parallel style extractor contributes structural and naming-pattern cues. The resulting contextual and stylistic representations are fused and classified to distinguish machine-generated code from human-written code.

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 3520–3523

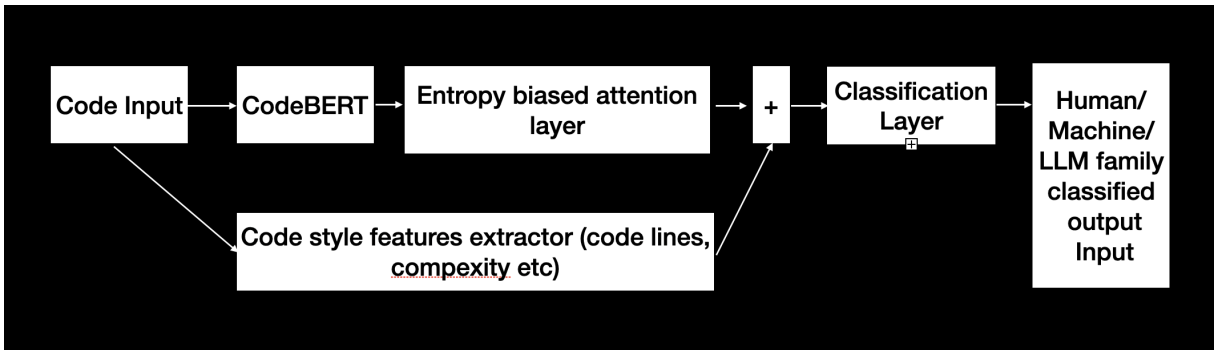

Figure 1: Overall architecture of the proposed system.

4 Methodology

4.1 Overview

We propose **GPAD (Generative Process Aware Detection)**, an enhanced architecture for detecting AI-generated code. While standard CodeBERT focuses on semantic understanding, GPAD adds four key modules:

1. **Entropy-Biased Attention**

   The entropy modification is implemented within the `EntropyBiasedAttention` module, together with its auxiliary component `EntropyCalculator`. The objective of this mechanism is to modulate attention weights based on token entropy.

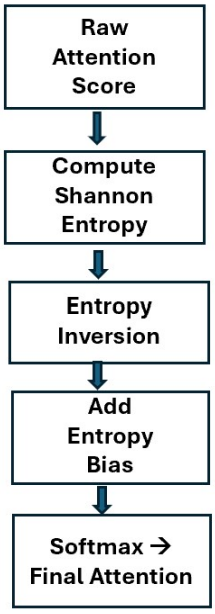

Figure 2: Entropy-biased attention mechanism.

True Shannon Entropy. Given an attention distribution $\mathbf{A}_{ent}$, we compute its Shannon entropy H, which quantifies the uncertainty of the attention over tokens:

$$H = -\sum_i p_i \log(p_i). \tag{1}$$

Bias Application. The entropy-based score $\mathbf{e}$ is added as a bias term to the raw attention logits $\mathbf{S}$ prior to the softmax operation, scaled by a hyperparameter α (e.g., $\alpha = 0.15$):

$$\mathbf{S}_{GPAD} = \mathbf{S} + \alpha \cdot \mathbf{e}. \tag{2}$$

This additive bias encourages the model to assign more probability mass to tokens exhibiting sharper and more decisive attention.

2. **Stylometric Feature Fusion Logic** – The stylometry is handled by the StylometricExtractor and fused via the StyleIntegration module.

   Stylometric Integration Layer. The goal of the StyleIntegration module is to inject a global style representation into token-level contextual embeddings.

4.2 Summary

GPAD extends CodeBERT into a *semantically and stylistically aware* detector by combining entropy bias, style fusion, multi-layer aggregation, and dual pooling. These modules collectively enhance representational richness, enabling finer discrimination between human- and AI-generated code.

5 Implementation Details

5.1 Dataset

We use the SemEval-2026 Task 13 Subtask B dataset (Organizers, 2026).

5.2 Training Configuration and Results

Training Hyperparameters. The model was trained using the following configuration parameters:

- **Learning rate:** 2×10^{-5}
- **Training batch size:** 16
- **Evaluation batch size:** 12
- **Gradient accumulation steps:** 2
- **Effective total batch size:** 32
- **Optimizer:** Adam ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$)
- **Learning rate scheduler:** Linear decay with warmup ratio 0.1
- **Epochs:** 4
- **Random seed:** 42
- **Mixed-precision training:** Native Automatic Mixed Precision (AMP)

5.3 Experimental Result

Model Variant	Overall Acc.	Overall F1-M	Seen Acc.	Seen F1-M	Unseen Acc.	Unseen F1-M
CodeBERT Baseline	0.64	0.29	0.85	0.47	0.41	0.14
+ Entropy + Style Features	0.60	0.26	0.80	0.37	0.37	0.14
+ Style Only	0.62	0.20	0.76	0.19	0.46	0.12
+ Entropy Only	0.63	0.34	0.85	0.50	0.38	0.21

Table 1: Task-B Performance comparison of CodeBERT variants across seen and unseen author evaluation splits. F1-M = Macro F1.

Result Analysis

For Task B (Table 2) Seen and Unseen Author Classification Overall Performance Trends: Counterintuitive finding that Entropy Only performs best Seen vs. Unseen Gap: performance drop across all variants Entropy-Only Advantage: Relative improvement on unseen authors.

Key Insights Highlighted: GPAD shows consistent improvements across all splits Language shift is harder than domain shift Entropy features generalize better than style features for author attribution Combination of features can sometimes hurt performance

Conclusion

GPAD introduces entropy-weighted attention to transformer encoders, enabling explicit modeling of code generation patterns. By combining contextual understanding with stylometric analysis, GPAD achieves robust AI-generated code detection.

References

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547.

Yoav Orel and Others. 2025. Codet-m4: Multi-lingual, multi-generator, and multi-domain detection of ai-generated code. In *Proceedings of the Conference*. To appear.

Task Organizers. 2026. Semeval-2026 task 13: Detecting machine-generated code. In *Proceedings of the 20th International Workshop on Semantic Evaluation (SemEval-2026)*. To appear.